TESTE TÉCNICO DEV BACKEND

A Cotabest é um Market Place que oferece ao comprador diversas opções de compra, variando entre fornecedores e disponibilidade dos produtos em forma de atacado.
Considerando essas informações, desenvolva um sistema utilizando Django, Flask ou FastAPI que:

1 - Forneça um endpoint de busca de produtos pelo nome.
- A busca deve considerar apenas o nome do produto, sem diferenciar maiúsculas e minusculas.
- A acentuação pode ser considerada neste desafio, ou seja: o produto RAÇÃO não precisa ser retornado em uma busca por RACAO, porém qualquer parte do nome deve ser considerada (R, RA, AÇAO, RAÇÃ, etc)
- O Endpoint deve retornar um json como abaixo:

```
[
      {
            "id": 1,
            "name": "Ração para cachorro",
            "price": 50.00,
            "minimun": 10,
            "amount-per-package": 2,
            "max-availability": 50000
      },
      {
            "id": 7,
            "name": "Ração para gato",
            "price": 50.00,
            "minimun": 40,
            "amount-per-package": 7,
            "max-availability": 50000
      },
]
```

2 - Crie um sistema de carrinho e disponibilize os endpoints onde um usuário possa:
- adicionar um produto ao carrinho, informando a quantidade desejada
- alterar a quantidade de um determinado produto no carrinho
- remover um produto do carrinho
- exibir todos os produtos no carrinho com quantidade e valor total

3 - disponibilize um endpoint onde o usuário consiga finalizar a compra
- caso todos os produtos atendam a quantiidade mínima (minimum) o sistema deve gerar um id de pedido e zerar o carrinho.
- A resposta do endpoint deve ser como:

```
{
       "id": "cecbdc8e-23b8-45ad-a21f-0b803a700ba2"
      "total-price": 7780.00,
      "items": [
            {
                  "id": 1,
                  "name": "Ração para cachorro",
                  "price": 50.00,
                  "minimun": 10,
                  "amount-per-package": 2,
                  "quantity": 150
            },
            {
                  "id": 7,
                  "name": "Ração para gato",
                  "price": 50.00,
                  "minimun": 40,
                  "amount-per-package": 7,
                  "quantity": 56
            },
      ]
}
```

Observações:
1 - caso a quantidade de itens de algum dos produtos não atinja a quantidade mínima, o sistema deve responder com uma mensagem informando o usuário sobre a quantidade mínima para aquele produto;
2 - ao adicionar ou editar a quantidade de um produto é obrigatório verificar se o valor é compatível com a quantidade por pacote (amount per package) - exemplo: refrigerante é vendido em fardos, mas o valor é unitário e no atacado o fardo é inviolável
4 - A quantidade adicionada ao carrinho não pode ultrapassar a quantidade em estoque (max-availability)
3 - Ao finalizar a compra o carrinho deve ser resetado.
4 - Não será avaliada a utilização de base de dados no teste, podendo as informações serem armazenada no DB de sua preferência ou no sistema de arquivos.
5 - O projeto deve ter pelo menos 80% de cobertura de testes unitários

6 - Organize o projeto da forma como você organizaria uma aplicação para ser entregue em produção
7 - Utilize os recursos e bibliotecas que achar necessários para resolver o teste. Isso fará parte da avaliação
8 - Disponibilize o projeto em seu repositório no Github e envie-nos o link

Anexo ao e-mail está um json que você pode usar como base de dados para este teste.